# Burak Tekdamar
# 161044115
# CSE 344 FINAL REPORT

## 1 Overview

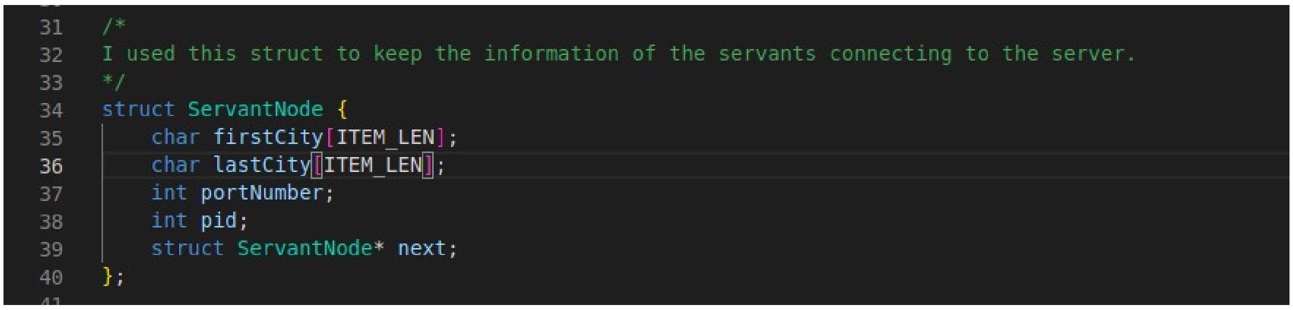
First of all, I planned the structures that I need to use in server, servant and client separately. In the server, I used a queue structure to queue up the requests coming from the clients and a linked list structure to keep the information of the open servants. On the client, I read the requests in the file line by line and created a thread in each request. I also enabled the threads to send requests to the server at the same time by using the barrier. On the Servant side, I kept the file contents of all the cities that should be interested in according to the range value entered by the user in a linked list structure. I defined the structure I used as a nested linked list. In other words, I kept the files in the cities as a separate list in the list where I kept the cities, and the data in the files as a separate list in the file list.

## 2 How did I solve this

### Server

First of all, I parse the values entered by the user from the terminal with the getopt() function. When an incorrect entry is made, I return an error and terminate the program.

The server initially creates a shared memory for the servants, writes its own port here, and creates a semaphore for the servants to reach this shared memory in a synchronized manner. It creates a queue structure to collect incoming requests. It opens a socket to process the data coming from the client and the servant. In order to separate the requests from the client and the servant, I used the letter "S" at the beginning of the requests from the servant and the letter "C" at the beginning of the requests from the client. If a request comes from a servant, after parsing it, I put the information in the servant list, and if it comes from a client, I put it in the queue. After a request arrives, a signal is sent to the condition variable in the threads and the thread is unlocked to handle that request. The thread first keeps the fd value of the socket at the local level when the request arrives. then it checks if queue is empty. If it is empty, it receives the request from the queue and parses the information. If there is city information in the request, it sends the request to the port of the servant who is interested in that city. If the city information is not given, it sends a request to all open servants one by one and waits for an answer. The server continues its operations until the SIGINT signal is received. When SIGINT comes, it closes all open sources and sends this signal to the servants which are open.



```c
/*
I used this struct to keep the information of the servants connecting to the server.
*/
struct ServantNode {
    char firstCity[ITEM_LEN];
    char lastCity[ITEM_LEN];
    int portNumber;
    int pid;
    struct ServantNode* next;
};
```

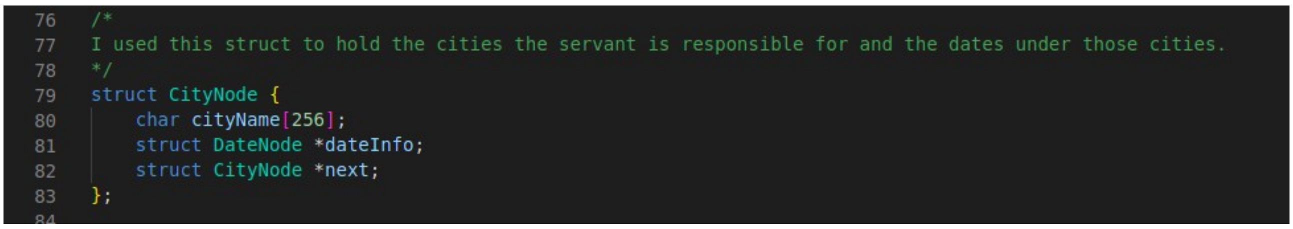
***Servant list node struct on server***

## Servants

First of all, I parse the values entered by the user from the terminal with the getopt() function. When an incorrect entry is made, I return an error and terminate the program.

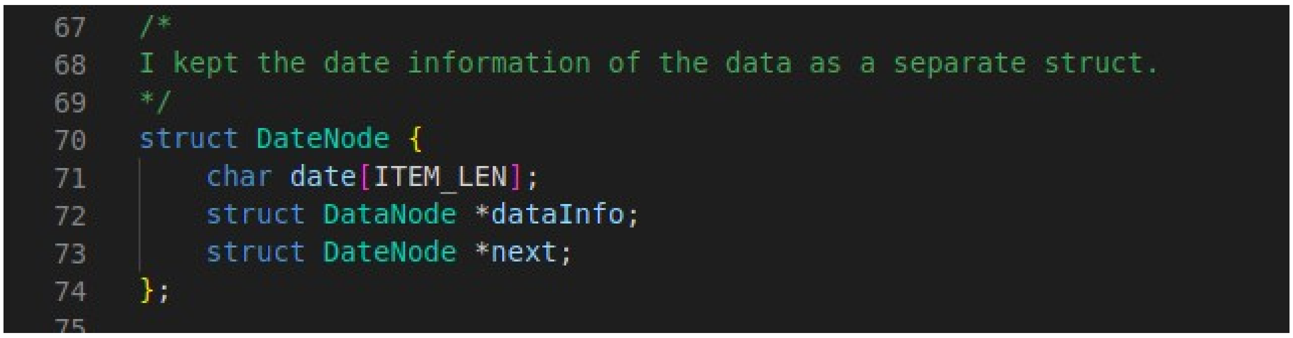
Then I created the socket that the servant will use and receive a request from the server and send a response. Then I opened the shared memory and semaphore that I opened on the server side to use it. Then I locked the semaphore and read the port number written in the shared memory block and increased it by one. I tried connecting to the port by calling the bind function. If the bind function returns a negative value, I increased the port number again by one. After bind was successful, I wrote the new port value to the shared memory for other servants. All servants set their own port values with this method. Then, each servant reads the information of the cities they will handle with the scandir() function and sends a linked list. First, the folders containing the cities are browsed one by one and the names of these folders are added to the list if they are within the range given by the user. After the city is found, the folder of that city is scanned and the dates in it are kept as a list in the city linked list. The information in each date file is also kept as a data list within the date list. Then, each servant sends his port number and the city range he handles to the server. From here on, the communication to be established with the server will be made over the port belonging to the servant.

After each request to the servants, a thread will be created and the incoming request will be searched among the data. Then the number found will be sent to the server as a response and the program will continue to wait for the new request. I'm reading the "/proc/self" symbolic link path to find pid of servants.



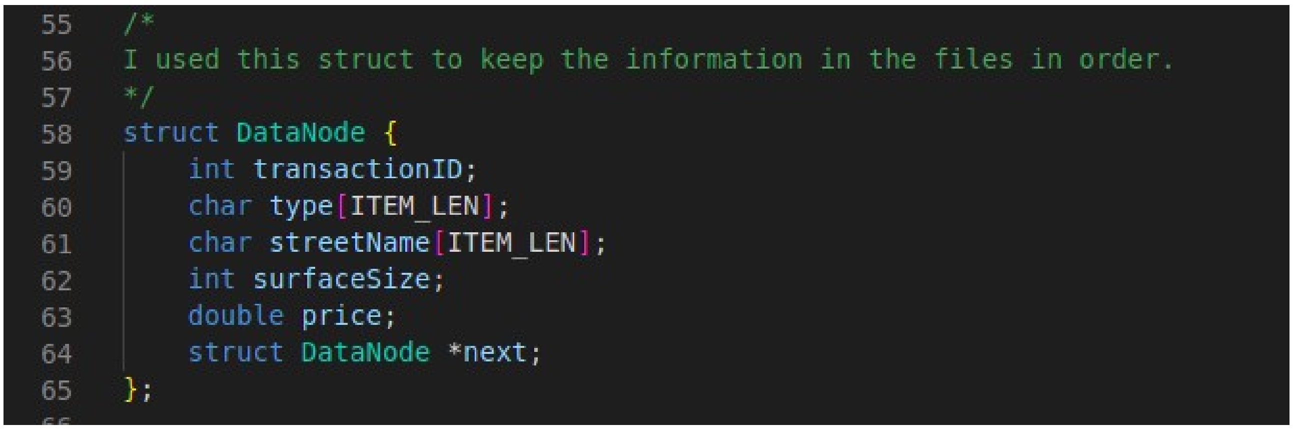
```c
/*
I used this struct to hold the cities the servant is responsible for and the dates under those cities.
*/
struct CityNode {
    char cityName[256];
    struct DateNode *dateInfo;
    struct CityNode *next;
};
```

***City list node structure***



```c
/*
I kept the date information of the data as a separate struct.
*/
struct DateNode {
    char date[ITEM_LEN];
    struct DataNode *dataInfo;
    struct DateNode *next;
};
```

***Date list node structure***



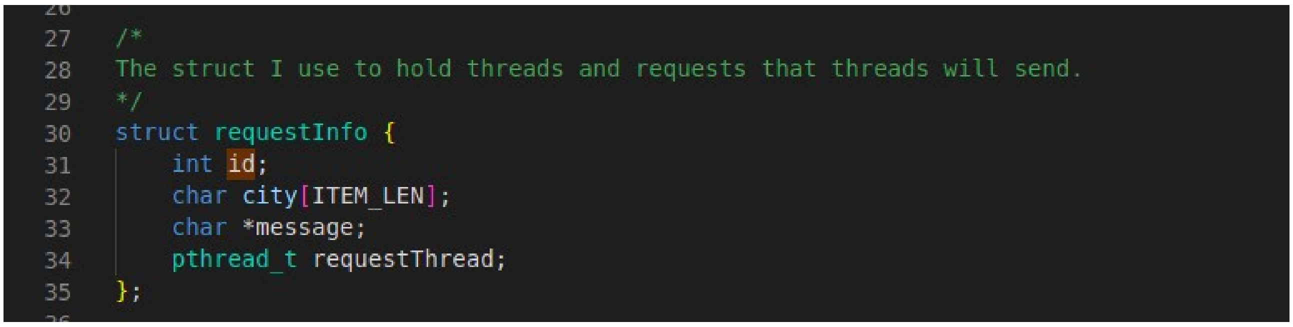
```c
/*
I used this struct to keep the information in the files in order.
*/
struct DataNode {
    int transactionID;
    char type[ITEM_LEN];
    char streetName[ITEM_LEN];
    int surfaceSize;
    double price;
    struct DataNode *next;
};
```

***Data list node structure***

## Client

First of all, I parse the values entered by the user from the terminal with the getopt() function. When an incorrect entry is made, I return an error and terminate the program.

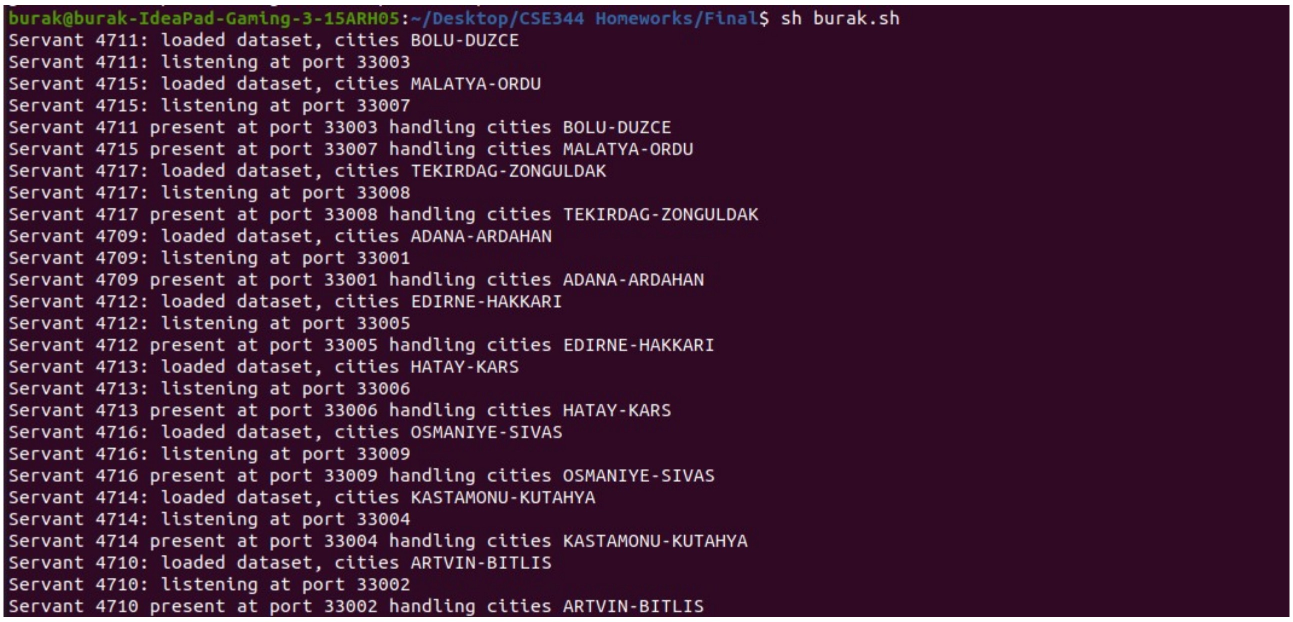
By reading the given request file line by line, I put each request in a struct and added these structs to a linked list. Each request has its own pthread variable. After the file reading process is finished, the threads added to the list are created. There is a barrier inside the thread function. When the number of threads created reaches the total number of requests, the pthread_cond_broadcast() function is called and sending operations to the server are started. The thread that receives a response from the server terminates. The client terminates after all threads receive a reply.



```c
/*
The struct I use to hold threads and requests that threads will send.
*/
struct requestInfo {
    int id;
    char city[ITEM_LEN];
    char *message;
    pthread_t requestThread;
};
```

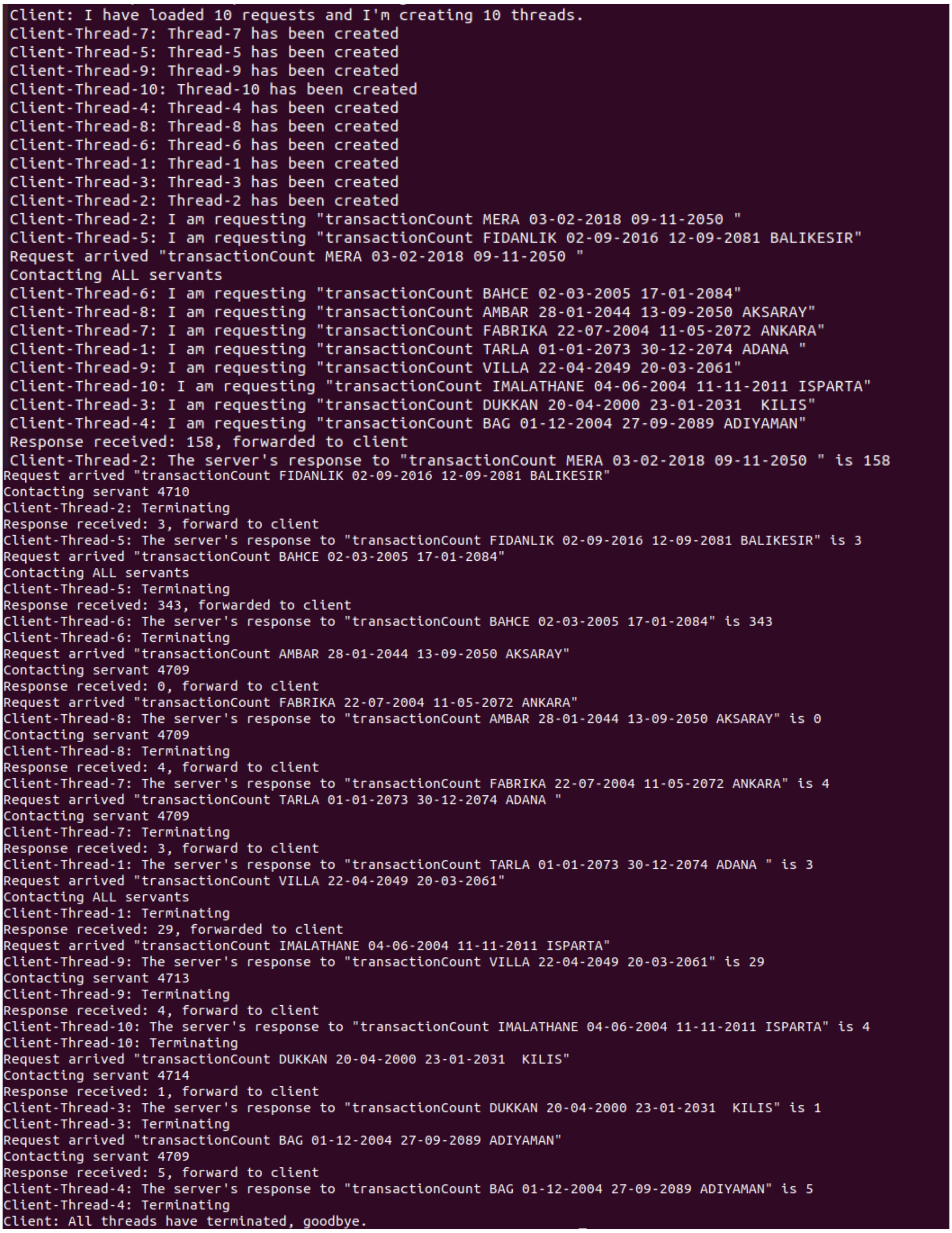
***Request info structure***

## OUTPUTS



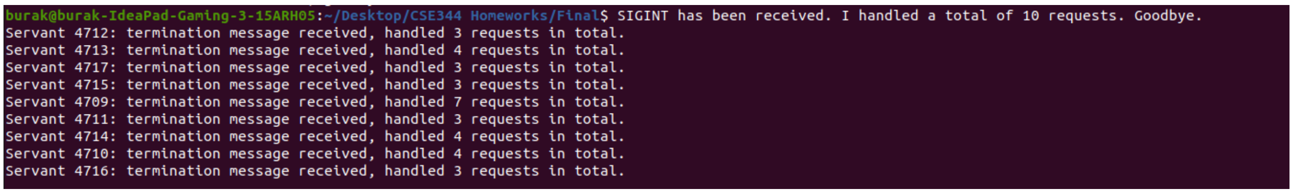
```
burak@burak-IdeaPad-Gaming-3-15ARH05:~/Desktop/CSE344 Homeworks/Final$ sh burak.sh
Servant 4711: loaded dataset, cities BOLU-DUZCE
Servant 4711: listening at port 33003
Servant 4715: loaded dataset, cities MALATYA-ORDU
Servant 4715: listening at port 33007
Servant 4711 present at port 33003 handling cities BOLU-DUZCE
Servant 4715 present at port 33007 handling cities MALATYA-ORDU
Servant 4717: loaded dataset, cities TEKIRDAG-ZONGULDAK
Servant 4717: listening at port 33008
Servant 4717 present at port 33008 handling cities TEKIRDAG-ZONGULDAK
Servant 4709: loaded dataset, cities ADANA-ARDAHAN
Servant 4709: listening at port 33001
Servant 4709 present at port 33001 handling cities ADANA-ARDAHAN
Servant 4712: loaded dataset, cities EDIRNE-HAKKARI
Servant 4712: listening at port 33005
Servant 4712 present at port 33005 handling cities EDIRNE-HAKKARI
Servant 4713: loaded dataset, cities HATAY-KARS
Servant 4713: listening at port 33006
Servant 4713 present at port 33006 handling cities HATAY-KARS
Servant 4716: loaded dataset, cities OSMANIYE-SIVAS
Servant 4716: listening at port 33009
Servant 4716 present at port 33009 handling cities OSMANIYE-SIVAS
Servant 4714: loaded dataset, cities KASTAMONU-KUTAHYA
Servant 4714: listening at port 33004
Servant 4714 present at port 33004 handling cities KASTAMONU-KUTAHYA
Servant 4710: loaded dataset, cities ARTVIN-BITLIS
Servant 4710: listening at port 33002
Servant 4710 present at port 33002 handling cities ARTVIN-BITLIS
```

***Servants and server outputs***



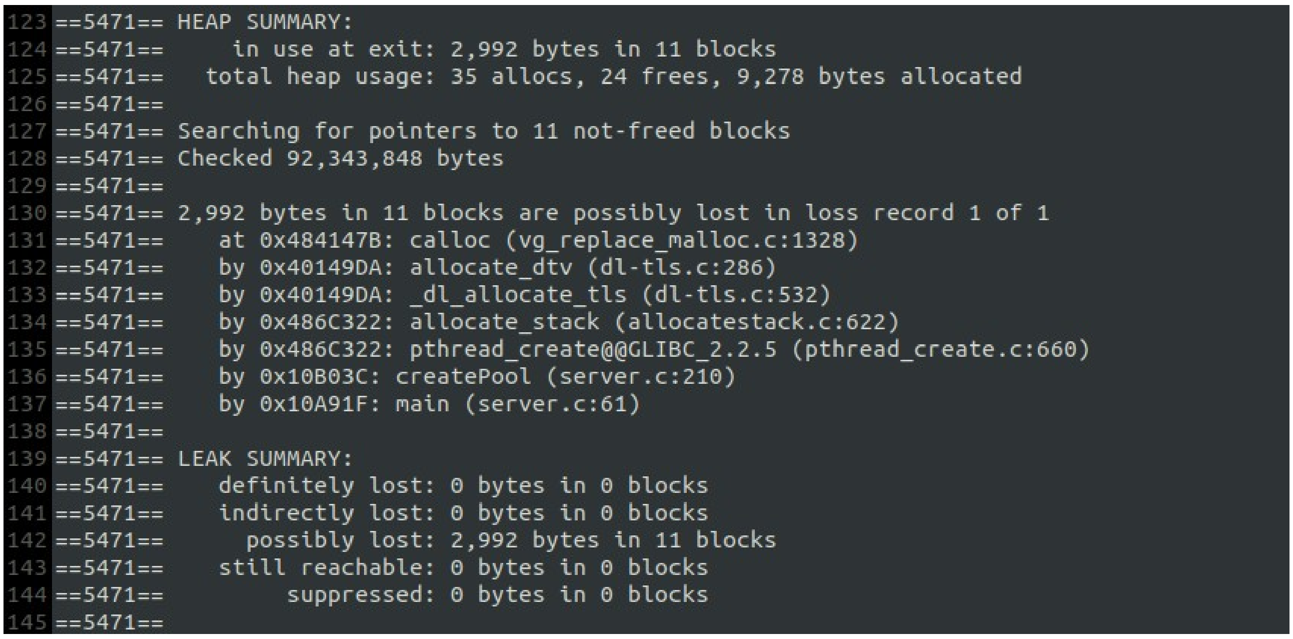
```
Client: I have loaded 10 requests and I'm creating 10 threads.
Client-Thread-7: Thread-7 has been created
Client-Thread-5: Thread-5 has been created
Client-Thread-9: Thread-9 has been created
Client-Thread-10: Thread-10 has been created
Client-Thread-4: Thread-4 has been created
Client-Thread-8: Thread-8 has been created
Client-Thread-6: Thread-6 has been created
Client-Thread-1: Thread-1 has been created
Client-Thread-3: Thread-3 has been created
Client-Thread-2: Thread-2 has been created
Client-Thread-2: I am requesting "transactionCount MERA 03-02-2018 09-11-2050 "
Client-Thread-5: I am requesting "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR"
Request arrived "transactionCount MERA 03-02-2018 09-11-2050 "
Contacting ALL servants
Client-Thread-6: I am requesting "transactionCount BAHCE 02-03-2005 17-01-2084"
Client-Thread-8: I am requesting "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY"
Client-Thread-7: I am requesting "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA"
Client-Thread-1: I am requesting "transactionCount TARLA 01-01-2073 30-12-2074 ADANA "
Client-Thread-9: I am requesting "transactionCount VILLA 22-04-2049 20-03-2061"
Client-Thread-10: I am requesting "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA"
Client-Thread-3: I am requesting "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS"
Client-Thread-4: I am requesting "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN"
Response received: 158, forwarded to client
Client-Thread-2: The server's response to "transactionCount MERA 03-02-2018 09-11-2050 " is 158
Request arrived "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR"
Contacting servant 4710
Client-Thread-2: Terminating
Response received: 3, forward to client
Client-Thread-5: The server's response to "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR" is 3
Request arrived "transactionCount BAHCE 02-03-2005 17-01-2084"
Contacting ALL servants
Client-Thread-5: Terminating
Response received: 343, forwarded to client
Client-Thread-6: The server's response to "transactionCount BAHCE 02-03-2005 17-01-2084" is 343
Client-Thread-6: Terminating
Request arrived "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY"
Contacting servant 4709
Response received: 0, forward to client
Request arrived "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA"
Client-Thread-8: The server's response to "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY" is 0
Contacting servant 4709
Client-Thread-8: Terminating
Response received: 4, forward to client
Client-Thread-7: The server's response to "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA" is 4
Request arrived "transactionCount TARLA 01-01-2073 30-12-2074 ADANA "
Contacting servant 4709
Client-Thread-7: Terminating
Response received: 3, forward to client
Client-Thread-1: The server's response to "transactionCount TARLA 01-01-2073 30-12-2074 ADANA " is 3
Request arrived "transactionCount VILLA 22-04-2049 20-03-2061"
Contacting ALL servants
Client-Thread-1: Terminating
Response received: 29, forwarded to client
Request arrived "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA"
Client-Thread-9: The server's response to "transactionCount VILLA 22-04-2049 20-03-2061" is 29
Contacting servant 4713
Client-Thread-9: Terminating
Response received: 4, forward to client
Client-Thread-10: The server's response to "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA" is 4
Client-Thread-10: Terminating
Request arrived "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS"
Contacting servant 4714
Response received: 1, forward to client
Client-Thread-3: The server's response to "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS" is 1
Client-Thread-3: Terminating
Request arrived "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN"
Contacting servant 4709
Response received: 5, forward to client
Client-Thread-4: The server's response to "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN" is 5
Client-Thread-4: Terminating
Client: All threads have terminated, goodbye.
```

***Requests output***



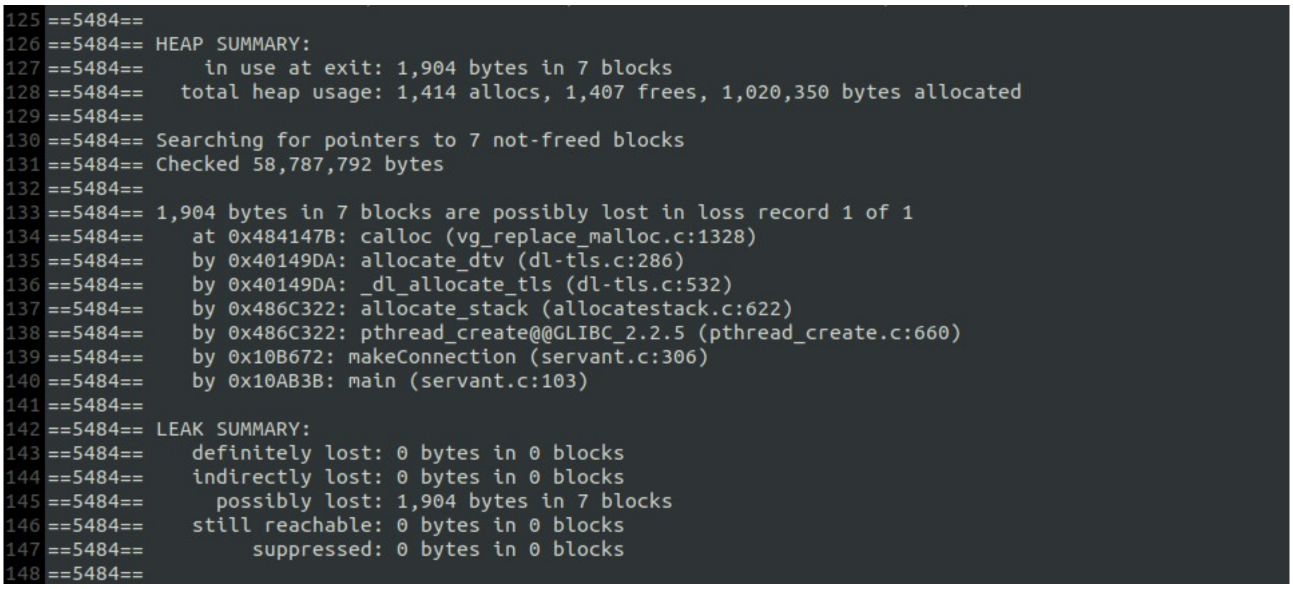
```
burak@burak-IdeaPad-Gaming-3-15ARH05:~/Desktop/CSE344 Homeworks/Final$ SIGINT has been received. I handled a total of 10 requests. Goodbye.
Servant 4712: termination message received, handled 3 requests in total.
Servant 4713: termination message received, handled 4 requests in total.
Servant 4717: termination message received, handled 3 requests in total.
Servant 4715: termination message received, handled 3 requests in total.
Servant 4709: termination message received, handled 7 requests in total.
Servant 4711: termination message received, handled 3 requests in total.
Servant 4714: termination message received, handled 4 requests in total.
Servant 4710: termination message received, handled 4 requests in total.
Servant 4716: termination message received, handled 3 requests in total.
```

***Termination results***



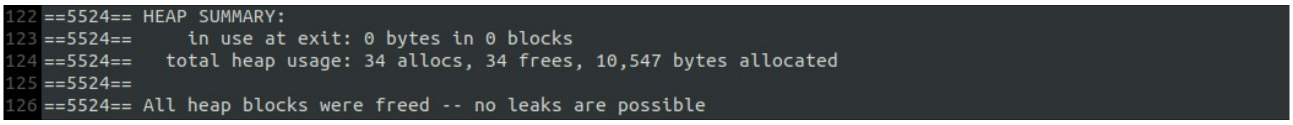
```
==5471== HEAP SUMMARY:
==5471==     in use at exit: 2,992 bytes in 11 blocks
==5471==   total heap usage: 35 allocs, 24 frees, 9,278 bytes allocated
==5471==
==5471== Searching for pointers to 11 not-freed blocks
==5471== Checked 92,343,848 bytes
==5471==
==5471== 2,992 bytes in 11 blocks are possibly lost in loss record 1 of 1
==5471==    at 0x484147B: calloc (vg_replace_malloc.c:1328)
==5471==    by 0x40149DA: allocate_dtv (dl-tls.c:286)
==5471==    by 0x40149DA: _dl_allocate_tls (dl-tls.c:532)
==5471==    by 0x486C322: allocate_stack (allocatestack.c:622)
==5471==    by 0x486C322: pthread_create@@GLIBC_2.2.5 (pthread_create.c:660)
==5471==    by 0x10B03C: createPool (server.c:210)
==5471==    by 0x10A91F: main (server.c:61)
==5471==
==5471== LEAK SUMMARY:
==5471==    definitely lost: 0 bytes in 0 blocks
==5471==    indirectly lost: 0 bytes in 0 blocks
==5471==      possibly lost: 2,992 bytes in 11 blocks
==5471==    still reachable: 0 bytes in 0 blocks
==5471==         suppressed: 0 bytes in 0 blocks
==5471==
```

***Server valgrind results***



```
==5484==
==5484== HEAP SUMMARY:
==5484==     in use at exit: 1,904 bytes in 7 blocks
==5484==   total heap usage: 1,414 allocs, 1,407 frees, 1,020,350 bytes allocated
==5484==
==5484== Searching for pointers to 7 not-freed blocks
==5484== Checked 58,787,792 bytes
==5484==
==5484== 1,904 bytes in 7 blocks are possibly lost in loss record 1 of 1
==5484==    at 0x484147B: calloc (vg_replace_malloc.c:1328)
==5484==    by 0x40149DA: allocate_dtv (dl-tls.c:286)
==5484==    by 0x40149DA: _dl_allocate_tls (dl-tls.c:532)
==5484==    by 0x486C322: allocate_stack (allocatestack.c:622)
==5484==    by 0x486C322: pthread_create@@GLIBC_2.2.5 (pthread_create.c:660)
==5484==    by 0x10B672: makeConnection (servant.c:306)
==5484==    by 0x10AB3B: main (servant.c:103)
==5484==
==5484== LEAK SUMMARY:
==5484==    definitely lost: 0 bytes in 0 blocks
==5484==    indirectly lost: 0 bytes in 0 blocks
==5484==      possibly lost: 1,904 bytes in 7 blocks
==5484==    still reachable: 0 bytes in 0 blocks
==5484==         suppressed: 0 bytes in 0 blocks
==5484==
```

***Servant valgrind results***



```
==5524== HEAP SUMMARY:
==5524==     in use at exit: 0 bytes in 0 blocks
==5524==   total heap usage: 34 allocs, 34 frees, 10,547 bytes allocated
==5524==
==5524== All heap blocks were freed -- no leaks are possible
```

***Client valgrind results***